# **NOSE2 AE2 Report**

**How does each schedular work?**

**FCFS (non-preemptive) –** As soon as a process arrives, it is added to queue in that order. The schedular chooses the first process in the queue and executes it to completion. Hence average waiting times tend to be high, but it is the easiest to implement.

**SJF (non-preemptive) –** Processes are added to a queue and that queue is sorted in ascending order based on the service time of each process. Then the first process is selected to be executed to completion. In some scenarios can be quite efficient but is quite unrealistic.

**RR (Preemptive) –** It is the preemptive version of FCFS, processes are added to queue in order they come in. The first task is executed, but not to completion but for the duration of the time-slice, if the process ends before the time variable the process is terminated and the schedular moves onto the next task. Else the remaining process at the end of the queue.

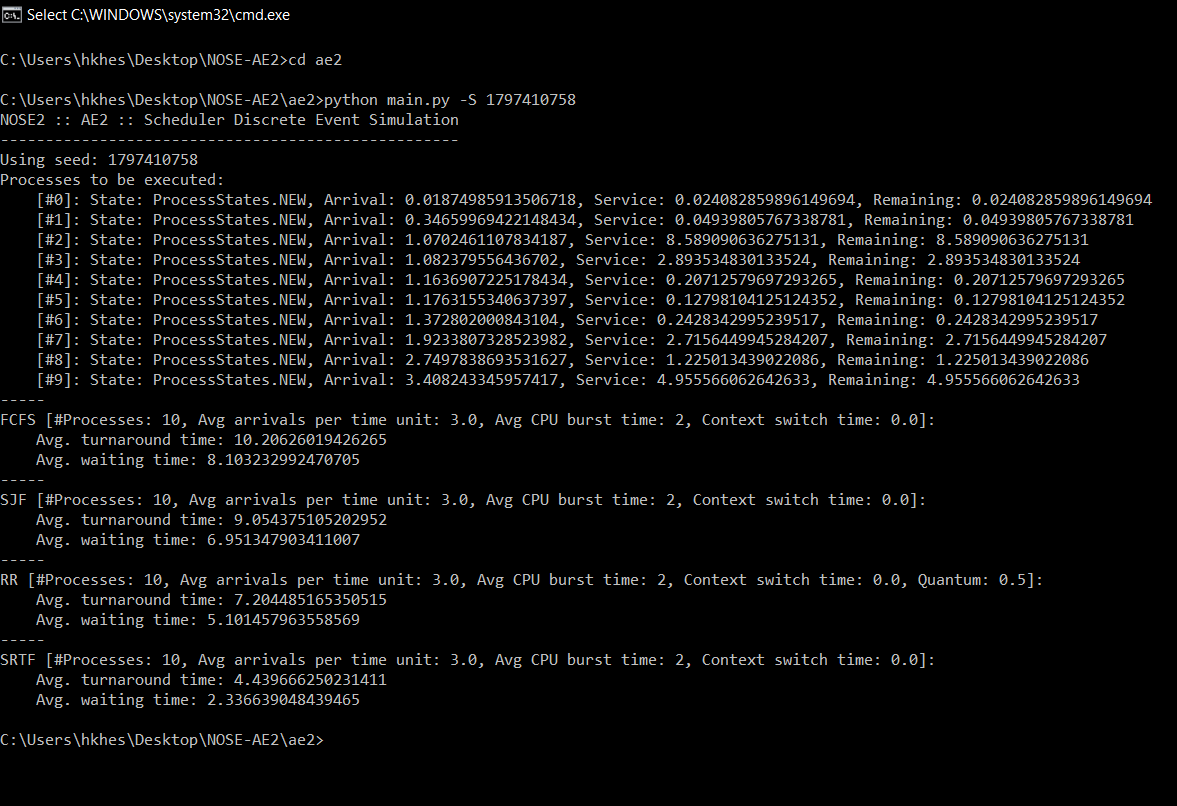
**SRTF (Preemptive)** – Is the preemptive version of the SJF, so the process in the queue are added ordered based on their remaining CPU burst time. The schedular is called every time a new process arrives. Executes like SJF, in term like it will not execute to completion, and if it is not finished it will be added back to the queue.

**Preemptive vs non – preemptive:**

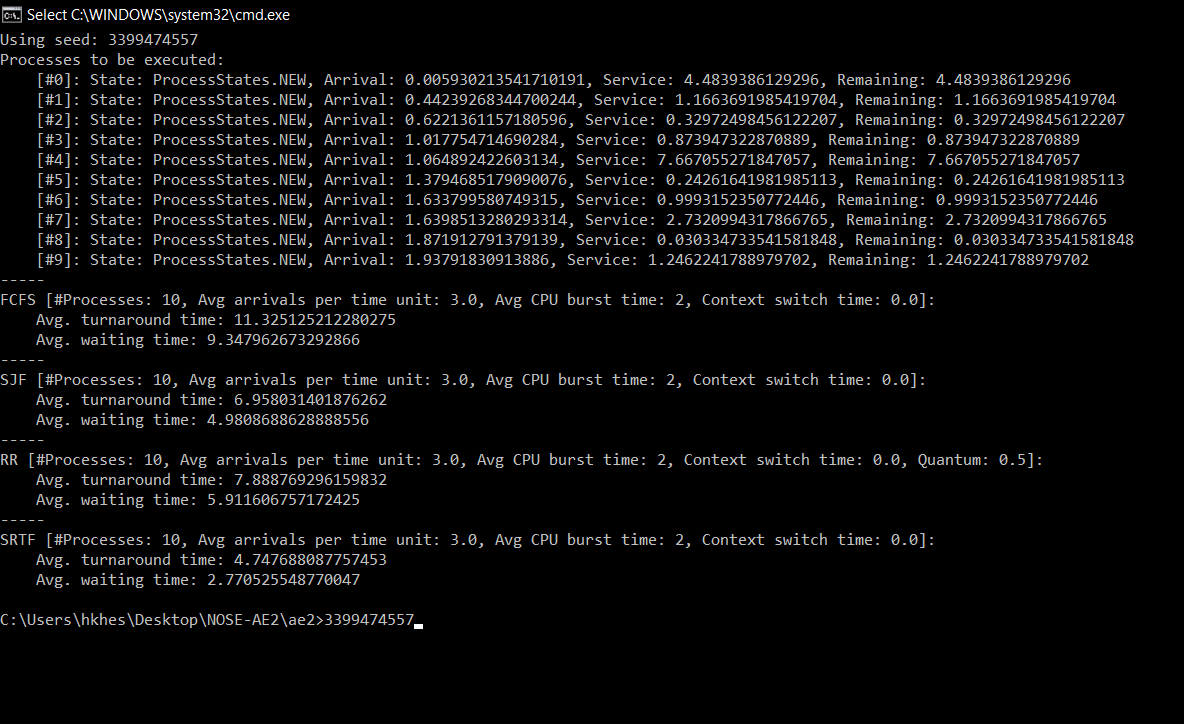
**Preemptive:** Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining.

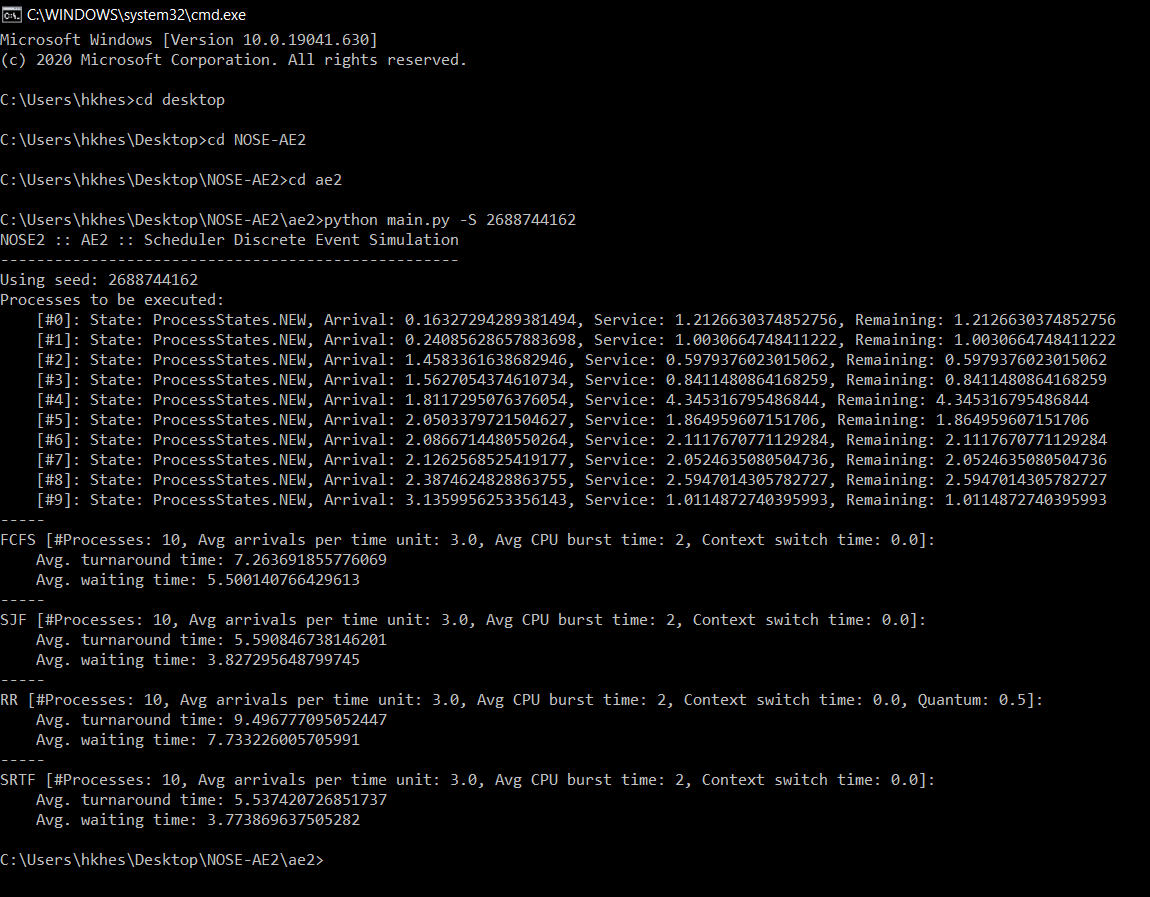
**Non – preemptive:** Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. Waits till the process complete its CPU burst time and then it can allocate the CPU to another process

**Seed Number: 1797410758**

From the results we can deduce that both preemptive schedulers have lower average turnaround and waiting times, mainly because the processes are around if the quantum. In between SRTF and RR, SRTF did significantly better, as the SRTF sorts the files, resulting in a lower wait time as all the small task are being done first and not being held up by the larger tasks. RR’s quantum time is larger than the size processes, causing it to be slower than the SRTF.

**Seed Number: 3399474557**

Both the schedulers that order the queue, SJF and SRTF, in terms of service time and remaining time, respectively, outperformed both the schedulers that did not, FCFS and RR. This could mean that the SJF and SRTF receive mainly the sorted ones as they sort it by the shortest and runs them. In the case of the other two if a large process comes in it will run it making the shorter task wait, that is why the turn around and time is greater. RR has a shorter avg. turnaround and waiting than FCFS as the processes size is greater than the quantum.

**Seed Number: 2688744162**

This seed number shows a similar trend to the “339947557”, where SRTF and SJF did the best, but this time the difference between SRTF and SJF is smaller, as SRTF sorts by remaining time it finishes the quicker ones first, hence reducing the Avg. turnaround and waiting time ever so slightly. The time out of the schedulers that don’t sort the processes, FCFS performed better than RR, the processes are shorter than the quantum time for RR, hence RR would wait for the quantum time to finish hence take more time to move on to the next process, resulting in a higher Avg. turnaround and waiting time.

In conclusion, throughout the 3 examples, the SRTF, consistently performs the best. As a mixture of being pre-emptive and as it sorts the list, it has the best of both worlds. FCFS being the simplest to implement has consistently highest avg. turnaround and waiting time. Depending on the processes size, SJF will perform better as it has an ordered list getting rid of the short tasks, leading to lower avg. waiting and turnaround time. Or RR will do better as it is pre-emptive, so when the processes size is grater than the quantum time as that would lead to lower avg. turnaround and waiting time.